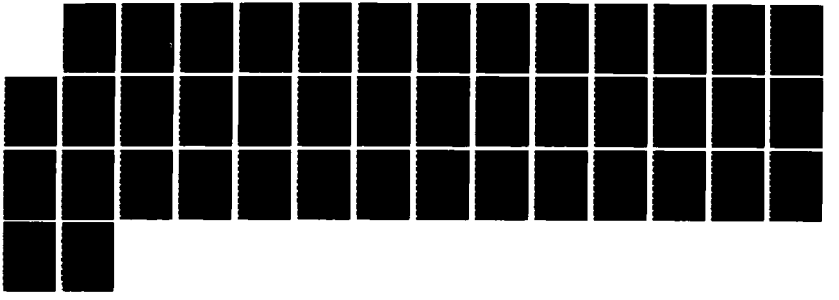
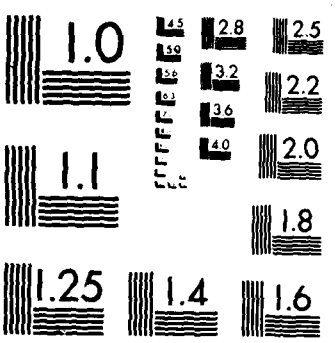
AD-A174 988 ADA COMPILER VALIDATION SUMMARY REPORT: TELELOGIC 1/1
TELESOFT ADA VERSION 376 SUN 2-120(U)
INDUSTRIEANLAGEN-BETRIEBSGESELLSCHAFT M B H OTTOBRUNN
UNCLASSIFIED (GERMANY F R) 21 APR 86 F/G 9/2 NL





# Ada* COMPILER VALIDATION SUMMARY REPORT:
TeleLOGIC
TeleSoft Ada, Version 3.7
Sun 2-120

Completion of On-Site Validation:
86-04-21

Prepared By:
IABG m.b.H., Dept SZT
Einsteinstrasse 20
D 8012 Ottobrunn

Prepared For:
Ada Joint Program Office
United States Department of Defense
Washington, D.C.



* Ada is a registered trademark of the United States Government (Ada Joint Program Office)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETEING FORM

| | |
|---|---|
| 1. REPORT NUMBER \| 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Ada Compiler Validation Summary Report:<br>TeleLOGIC Telesoft Ada, Version 3.7 | 5. TYPE OF REPORT & PERIOD COVERED<br>April 1986 to April 1987<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>IABG m.b.h., Dept. SZT<br>Einsteinstrasse 20<br>D 8012 Ottobrunn | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION AND ADDRESS<br>IABG M.B.H., Dept. SZT | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Ada Joint Programming Office<br>United States Department of Defense<br>Washington, D.C. 20301-3081 | 12. REPORT DATE<br>21 April 86<br>13. NUMBER OF PAGES<br>40 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>IABG m.b.h., Dept. SZT<br>Einsteinstrasse 20<br>D 8012 Ottobrunn | 15. SECURITY CLASS (of this report)<br>UNCLASSIFIED<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Ada Programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, ACVC, Validation Testing, Ada Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD-1815A, Ada Joint Program Office, AJPO

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Attached.

Ada* Compiler Validation Summary Report:

Compiler Name: TeleSoft_Ada, Version 3.7

| Host Computer | Target Computer |
|---|---|
| Sun 2-120 under Sun 2.0 | Sun 2-120 under Sun 2.0 |

Testing Completed Using ACVC 1.7

This report has been reviewed and approved:

IABG m.b.H., Dept SZT
Dr. Helmut Hummel
IABG, Dept SZT
Einsteinstrasse
D-8012 Ottobrunn

Ada Validation Office (AVO)
Dr. John F. Kramer
Institute for Defense Analyses
Alexandria, VA

Ada Joint Program Office (AJPO)
Virginia L. Castor
Director
Washington, D.C.



A-1

* Ada is a registered trademark of the United States Government (Ada Joint Program Office)

## EXECUTIVE SUMMARY

The Validation Summary Report presents the results and conclusions of testing performed on the TeleSoft_Ada, Version 3.7. Standardized tests serve as input to an Ada compiler, producing results which are evaluated by the validation team. This summary briefly states the highlights of the TeleSoft_Ada, Version 3.7, validation.

On-site testing was performed 86-04-11 through 86-04-21 at Nynäshamn under the auspices of the IABG m.b.H., Dept SZT (AVF), according to Ada Validation Office policies and procedures. The TeleSoft_Ada, Version 3.7, is hosted on Sun 2-120 operating under Sun 2.0. The suite of tests known as the Ada Compiler Validation Capability (ACVC), Version 1.7, was used. The ACVC is used to validate conformance of a compiler to ANSI/MIL-STD-1815A Ada. The purpose of testing is to ensure that a compiler properly implements legal language constructs and that it identifies and rejects illegal language constructs. The testing also identifies behavior that is implementation dependent but permitted by the Ada Standard. Six classes of tests are used. These tests are designed to perform checks at compile time, at link time, or during execution.

The results of validation are summarized in the following table.

| RESULT | TEST CLASS | | | | | | TOTAL |
|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | L | |
| Passed | 66 | 821 | 1023 | 17 | 9 | 21 | 1957 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 2 | 3 | 297 | 0 | 2 | 2 | 306 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 4 | 12 | 0 | 0 | 0 | 16 |
| TOTAL | 68 | 828 | 1332 | 17 | 11 | 23 | 2279 |

Tests found to contain errors were withdrawn from Version 1.7 of the Ada Compiler Validation Capability (ACVC). When validation was completed, the tests listed in Chapter 2.2 had been withdrawn.

One additional test had been withdrawn when prevalidation was completed. This is taken into account in the final tables.

Some tests demonstrate that language features are not supported by an implementation. For this implementation the tests determined the following.

. SHORT_INTEGER is not supported:
B52004E-AB.DEP B55B09D-AB.DEP B86001CR-AB.DEP
C34001D-B.DEP C55B07B-AB.DEP

. LONG_INTEGER is supported:
B52004D-AB.DEP B55B09C-AB.DEP B86001CS-AB.DEP
C34001E-B.DEP C55B07A-AB.DEP

. SHORT_FLOAT is not supported:
B86001CP-AB.DEP C34001F-B.DEP C35702A-AB.DEP

. LONG_FLOAT is not supported:

B86001CQ-AB.DEP C34001G-B.DEP C35702B-AB.DEP

. Representation specifications for noncontiguous enumeration representations are not allowed:

C55B16A-AB.DEP
B86001DT-AB.TST

. The only predefined numeric types are INTEGER, FLOAT, LONG_INTEGER:

. The package SYSTEM is used by package TEXT_IO:

C86001F-B.DEP

. The 'SIZE clause is not supported:

C87B62A-B.DEP

. The 'STORAGE_SIZE clause is not supported:

C87B62B-B.DEP

. The 'SMALL clause is not supported:

C87B62C-B.DEP

. Generic subroutine declarations and bodies cannot be compiled in separate compilation units:

CA1012A-B.DEP

. Generic package bodies cannot be compiled in separate compilation files:

CA2009C-B.DEP

. Generic subprogram bodies cannot be compiled in separate compilation files:

    CA2009F-B.DEP

. Pragma INLINE is not supported for procedures:

    LA3004A-AB.ADA  CA3004E-B.ADA
    E13004C-B.ADA

. Pragma INLINE is not supported for functions:

    LA3004B-AB.ADA  CA3004F-B.ADA
    EA3004D-B.ADA

. Mode IN_FILE is supported (for sequential I/O):

    CE2102D-B.ADA

. Mode OUT_FILE is supported (for sequential I/O):

    CE2102E-B.ADA

. Mode INOUT_FILE is supported (for direct I/O):

    CE2102F-B.ADA

. Mode RESET and DELETE are supported (for sequential and direct I/O):

    CE2102G-B.ADA

. Mode IN_FILE is supported (for direct I/O):

    CE2102I-B.ADA

. Mode OUT_FILE is supported (for direct I/O):

    CE2102J-B.ADA

. Dynamic creation and deletion of files are allowed:

    CE2106A-B.DEP  CE3110A-B.ADA

. More than one internal file can be associated with the same external file:

    CE2107B-B.ADA  CE2107C-B.ADA
    CE2107D-B.ADA  CE2111D-B.ADA
    CE3111B-B.ADA  CE3111C-B.ADA
    CE3114B-B.ADA

. Instantiation of package SEQUENTIAL_IO with

unconstrained array types is not allowed:

CE2201D-B.DEP

- Instantiation of package SEQUENTIAL_IO with unconstrained record types with discriminants is not allowed:

  CE2201E-B.DEP

- Dynamic creation and resetting of files is supported:

  CE2210A-B.ADA

- Instantiation of package DIRECT_IO with unconstrained array types and unconstrained types with discriminants is not supported:

  CE2401D-B.DEP

- An external file associated with more than one internal file can be reset:

  CE3115A-B.DEP

- Illegal filenames can exist:

  CE2102C-B.DEP

ACVC Version 1.7 was taken on-site via magnetic tape to Nynäshamn. The tape was loaded, and all tests, except for the executable tests which make use of a floating point precision greater than SYSTEM.MAX_DIGITS, were compiled on Sun 2-120. Class A, C, D, and E tests were executed on Sun 2-120.

On completion of testing, all results were analyzed for failed Class A, C, D, or E programs, and all Class B and L compilation results were individually analyzed.

The ACVC, Version 1.7, contains 2279 tests of which 1957 were applicable to TeleSoft_Ada, Version 3.7. No anomalies were found in the testing of this compiler. Testing demonstrated that all applicable tests were passed by this compiler and conformed to the Ada Standard. The AVF concluded that the results show acceptable compliance to ANSI/MIL-STD-1815A Ada.

## TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

The Validation Summary Report describes how an Ada compiler conforms to the language standard. This report explains all technical terms used within and thoroughly reports the Ada Compiler Validation Capability (ACVC) test results. Ada compilers must be written according to the language specification as given in the ANSI/MIL-STD-1815A Ada. All implementation-defined features must be included for the compiler to conform to the Standard. Following the guidelines of the Standard ensures continuity between compilers. That is, the entire Standard must be implemented, and nothing can be implemented that is not in the Standard.

Even though all validated Ada compilers conform to the Standard, it must be understood that some differences do exist between implementations. ANSI/MIL-STD-1815A permits some implementation dependencies, e.g., the maximum length of identifiers, the maximum values of integer types, etc. These implementation-dependent features limit the portability of programs between compilers. Other differences between compilers are due to limitations imposed on a compiler by the operating system and by the hardware. All of these dependencies are given in the report.

Validation summary reports are written according to a standardized format. Compiler users can, therefore, more easily compare the reports from several compilers when selecting a compiler for a given task. The validation report can be completed mostly from the test results produced during validation testing. Additional testing information is given at the end of the report and states problems and details which are unique for a specific compiler. The format of the validation report limits variance between reports, enhances readability of the report, and accelerates report readiness.

## 1 INTRODUCTION

### 1.1 Purpose of this Validation Summary Report

The Validation Summary Report documents the results of the testing performed on an Ada compiler. Testing was carried out for the following purposes:

. To identify any language constructs supported by the translator that do not conform to the Ada Standard

- To identify any unsupported language constructs required by the Ada Standard

- To describe the implementation-dependent behavior allowed by the Ada Standard

Testing of this compiler was conducted by IABG m.b.H., Dept SZT according to policies and procedures established by the Ada Validation Office (AVO). Testing was conducted from 86-04-11 through 86-04-21 at Nynäshamn.

### 1.2 Use of this Validation Summary Report

Consistent with the national laws of the originating country, the Ada Validation Office may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organizations represented on the signature page of this report do not represent or warrant that any statement or statements set forth in this report are accurate or complete, or that the subject compiler has no nonconformances to the Ada Standard other than those presented. This report is not intended for the purpose of publicizing the findings summarized herein.

Questions regarding this report or the validation tests should be directed to:

Ada Validation Office
Institute for Defense Analyses
1801 N. Beauregard
Alexandria VA 22311
U.S.A.

and to:

IABG m.b.H., Dept SZT
Einsteinstrasse 20
D-8012 Ottobrunn
Federal Republic of Germany

### 1.3 REFERENCES

1. Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A,

2. Ada Validation Organization: Policies and Procedures, MITRE Corporation, Jun 1982

3. Ada Compiler Validation Implementer's Guide, SofTech, Inc., Dec 1984.

## 1.4 DEFINITION OF TERMS

| | |
|---|---|
| Anomaly | A test result that, given pre-validation analysis, is not expected during formal validation but is judged allowable under the circumstances. |
| ACVC | The Ada Compiler Validation Capability. A set of programs that evaluates the conformance of a compiler to the Ada language specification, ANSI/MIL-STD-1815A. |
| Ada Standard | ANSI/MIL-STD-1815A, February 1983. |
| Applicant | The agency requesting validation. |
| AVF | The IABG m.b.H., Dept SZT. In the context of this report, the AVF is responsible for conducting compiler validations according to established policies and procedures. |
| AVO | The Ada Validation Office. In the context of this report, the AVO is responsible for setting policies and procedures for compiler validations. |
| Compiler | A processor for the Ada language. In the context of this report, a compiler is any language processor, including cross-compilers, translators, and interpreters. |
| Failed test | A test for which the compiler generates a result that demonstrates nonconformance to the Ada Standard. |
| Host | The computer on which the compiler resides. |
| Inapplicable test | A test that uses features of the language that a compiler is not required to support or may legitimately support in a way other than the one expected by the test. |
| Passed test | A test for which a compiler generates the expected result. |
| Target | The computer for which a compiler generates code. |

| | |
|---|---|
| Test | A program that evaluates the conformance of a compiler to a language specification. In the context of this report, the term is used to designate a single ACVC test. The text of a program may be the text of one or more compilations. |
| Withdrawn test | A test that has an invalid test objective, fails to meet its test objective, or contains illegal use of the language. |

## 1.5 Configuration

The candidate compilation system for this validation was tested under the configuration:

Compiler: TeleSoft_Ada, Version 3.7

Test Suite: Ada Compiler Validation Capability, Version 1.7

Host Computer:

| | |
|---|---|
| Two Machines: | Sun 2-120 |
| Operating System: | Sun 2.0 |
| Memory Size: | 3 MB |
| Disk System: | Sun 2.0 |

Target Computer:

Same as host computer

The two SUN computers are connected to a disk system, a VAX/UNIX, and a tape drive via Ethernet. One of the two SUN computers acts as a file server for both SUN computers. The VAX/UNIX system was used for printing of results only.

# CHAPTER 2

# TEST RESULTS

## 2.1 ACVC Test Classes

Conformance to ANSI/MIL-STD-1815A is measured using the Ada Compiler Validation Capability (ACVC). The ACVC contains both legal and illegal Ada programs structured into six test classes: A, B, C, D, E, and L. Legal programs are compiled and executed while illegal programs are just compiled. Support packages are used to report the results of the legal programs. A compiler must correctly process each of the tests in the suite and demonstrate conformance to the Ada Standard by either meeting the pass criteria given for the test or by showing that the test is inapplicable to the implementation. Tests that are found to contain errors are withdrawn from the ACVC. The results of validation testing are summarized in the following table:

| RESULT | TEST CLASS | | | | | | TOTAL |
|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | L | |
| Passed | 66 | 821 | 1023 | 17 | 9 | 21 | 1957 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 2 | 3 | 297 | 0 | 2 | 2 | 306 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 4 | 12 | 0 | 0 | 0 | 16 |
| TOTAL | 68 | 828 | 1332 | 17 | 11 | 23 | 2279 |

A total of 1993 tests were processed during this validation attempt, including 7 passed tests for the report package, 28 not applicable tests and 1 test withdrawn after 86-02-04. The 15 tests in Version 1.7 withdrawn until 86-02-04 were not processed, nor were 278 Class C tests that were inapplicable because they use floating point types having digits that exceed the maximum value for the implementation. All other tests were processed.

Some conventions are followed in the ACVC to ensure that the tests are reasonably portable without modification. For example, the tests make use of only the basic 55 character set, contain lines with a maximum length of 72 characters, use small numeric values, and place features that may not be supported in separate tests. However, some tests contain values that require the test to be customized according to implementation-specific values. The values used for this validation are listed in Appendix B.

### 2.1.1 Class A Tests

Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler. A Class A test is passed if no errors are detected at compile time and the program executes to produce a message indicating that it has passed. If a Class A test cannot be compiled and executed because of its size, then the test is split into a set of smaller subtests that can be processed. No splits were required.

The following table shows that all applicable Class A tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 15 | 9 | 0 | 5 | 2 | 12 | 13 | 3 | 0 | 0 | 0 | 7 | 66 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| TOTAL | 15 | 9 | 0 | 5 | 2 | 12 | 13 | 3 | 0 | 0 | 0 | 9 | 68 |

### 2.1.2 Class B Tests

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined manually to verify that every syntax or semantic error in the test is detected. A Class B test is passed if every illegal construct that it contains is detected by the compiler. If one or more errors are not detected, then a version of the test is created that contains only the undetected errors. The resulting "split" is compiled and examined. The splitting process continues until all errors are detected by the compiler. Splits were required for 11 tests:

| | | |
|---|---|---|
| B71001E | BA1101C4 | BA3013A6 |
| B71001Q | BA3006A6M | |
| B71001W | BA3006B3 | |
| B97101E | BA3007B7 | |
| | BA3008A4 | |
| | BA3008B5 | |

The following table shows that all applicable Class B tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 39 | 86 | 86 | 111 | 73 | 67 | 50 | 87 | 36 | 8 | 160 | 18 | 821 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 4 |
| TOTAL | 39 | 86 | 87 | 113 | 73 | 67 | 52 | 87 | 37 | 8 | 161 | 18 | 828 |

### 2.1.3 Class C Tests

Class C tests check that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASS/FAIL message indicating the result when it is executed. If a Class C test cannot be compiled because it exceeds the compiler's capacity, then the test is split into smaller subtests until all are compiled and executed. No splits were required.

The following table shows that all applicable Class C tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 37 | 89 | 162 | 117 | 82 | 18 | 93 | 109 | 39 | 20 | 56 | 201 | 1023 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 23 | 120 | 140 | 2 | 0 | 0 | 4 | 0 | 5 | 0 | 0 | 3 | 297 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 2 | 5 | 0 | 0 | 1 | 12 |
| TOTAL | 60 | 210 | 305 | 119 | 82 | 18 | 97 | 111 | 49 | 20 | 56 | 205 | 1332 |

### 2.1.4 Class D Tests

Class D tests check the compilation and execution capacities of a compiler. Since there are no requirements placed on a compiler by the Ada Standard for the number of identifiers permitted in a compilation, the number of units in a library, the number of nested loops in a subprogram body, and so on, a compiler may refuse to compile a Class D test. Each Class D test is self-checking and produces a PASS/FAIL message indicating the result when it is executed. If a Class D test fails to compile because the capacity of the compiler is exceeded, then the test is classified as inapplicable.

The following table shows that all applicable Class D tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 1 | 0 | 4 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| TOTAL | 1 | 0 | 4 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 |

Capacities measured by the Class D tests are detailed in section 2.4, IMPLEMENTATION CHARACTERISTICS.

### 2.1.5 Class E Tests

Class E tests provide information about the compiler in those areas in which the Ada Standard permits implementations to differ. Each Class E test is executable and produces messages that indicate how the Ada Standard is interpreted. However, in some cases the Ada Standard permits a compiler to detect a condition either at compile time or at execution time, and thus a Class E test may correctly fail to execute. A Class E test is passed if it fails to compile and appropriate error messages are issued, or if it executes properly and produces a message that it has passed. If a Class E test cannot be compiled and executed because of its size, then the test is split into a set of smaller subtests that can be processed. No splits were required.

The following table shows that all applicable Class E tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 1 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 9 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| TOTAL | 1 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 11 |

Information obtained from the Class E tests is detailed in section 2.4, IMPLEMENTATION CHARACTERISTICS.

### 2.1.6 Class L Tests

Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time and the test does not execute.

The following table shows that all applicable Class L tests passed:

| RESULT | CHAPTER | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 14 | TOTAL |
| Passed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 0 | 21 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inapplicable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 |
| Anomalous | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withdrawn | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| TOTAL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 23 |

### 2.1.7 Support Units

Three packages support the self-checking features of Class C tests: REPORT, CHECK_FILE, and VAR_STRINGS. The REPORT package provides the mechanism by which executable tests report results. It also provides a set of identity functions that are used to defeat some compiler optimization strategies to cause computations to be made by the target computer instead of the by the compiler on the host computer. The CHECK_FILE package is used to check the contents of text files written by some of the Class C tests for Chapter 14 of the Ada Standard. The VAR_STRINGS package defines types and subprograms for manipulating varying-length character strings. The operation of these three packages is checked by a set of executable tests. These tests produce messages that are examined manually to verify that the packages are operating correctly. If these packages are not operating correctly, then validation is not attempted.

An applicant is permitted to substitute the body of package REPORT with an equivalent one if for some reason the original version provided by the ACVC cannot be executed on the target computer. Package REPORT was modified for this validation in order to print the date and time of execution.

All support package specifications and bodies were compiled and were demonstrated to be operating correctly.

## 2.2 Withdrawn Tests

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. When preparing the tape for on site validation 15 tests had been withdrawn. When testing was performed another test had been withdrawn (C940ACA). Therefore, it was run as will but its result was ignored. In total, 16 tests had been withdrawn for the reasons indicated:

C35904A: The elaboration of subtype declarations SFX3 & SFX4 may raise NUMERIC_ERROR vs. CONSTRAINT_ERROR.

C41404A: The values of 'LAST and 'LENGTH in the "if" statements from line 74 to the end of the test are incorrect.

C48008A: This test requires that the evaluation of default initial values not occur if an exception is raised by an allocator. However, the LMC has ruled that such a requirement is incorrect (AI-00397).

B4A010C: The object_declaration in line 18 follows a subprogram body of the same declarative part.

C4A014A: The number declarations in lines 19-22 are not correct, because conversions are not static.

B83A06B: The Ada Standard 8.3(17) and AI-00330 permit the label LAB_ENUMERAL of line 80 to be considered a homograph of the enumeration literal in line 25.

C92005A: At line 40, "/=" for type PACK.BIG_INT is not visible without a "use" clause for package PACK.

C940ACA: This test assumes that allocated task TT1 will run prior to the main program, and thus assign SPYNUMB the value checked for by the main program; however, such an execution order is not required by the Ada Standard, so the test is erroneous.

CA1003B: This test requires all of the legal compilation units of a file containing some illegal units to be compiled and executed. But according to AI-00255 such a file may be rejected as a whole.

BA2001E: The Ada Standard 10.2(5) states that "simple names of all subunits that have the same ancestor library unit must be distinct identifiers." This test checks for the above condition when stubs are declared; but it is not clear that the check must be made then, as opposed to when the subunit is compiled.

CA3005A..D (4 tests): There exists no valid elaboration order for these tests.

BC3204C: The file BC3204C4 should contain the body for BC3204C0 --as indicated in line 25 of BC3204C3M.

CE2107E: TEMP_HAS_NAME must be given an initial value of TRUE.

## 2.3 Inapplicable Tests

Some tests do not apply to all compilers because they make use of features that a compiler is not required by the Ada Standard to support. Others may depend on the result of another test that is either inapplicable or withdrawn. For this validation attempt, 306 tests were inapplicable for the reasons indicated:

The testname indications ending in C-Y, respectively C-Z cover a number of 23, respectively 24 tests.

| | |
|---|---|
| AE2101C | Instantiation of I/O package with unconstrained array type not allowed |
| AE2101H | Instantiation of I/O package with unconstrained array type not allowed |
| B52004E | type SHORT_INTEGER not supported |
| B55B09D | type SHORT_INTEGER not supported |
| B86001DT | no valid macro expansion possible |
| C24113C-Y | Digits 7 - 29 |
| C34001D | type SHORT_INTEGER not supported |
| C34001F | type SHORT_FLOAT not supported |
| C34001G | type LONG_FLOAT not supported |
| C35702A | type SHORT_FLOAT not supported |
| C35702B | type LONG_FLOAT not supported |
| C35705C-Y | Digits 7 - 29 |
| C35706C-Y | Digits 7 - 29 |
| C35707C-Y | Digits 7 - 29 |
| C35708C-Y | Digits 7 - 29 |
| C35802C-Y | Digits 7 - 29 |
| C45241C-Y | Digits 7 - 29 |
| C45321C-Y | Digits 7 - 29 |
| C45421C-Y | Digits 7 - 29 |
| C45424C-Y | Digits 7 - 29 |
| C45521C-Z | Digits 7 - 30 |
| C45621C-Z | Digits 7 - 30 |
| C55B07B | type SHORT_INTEGER not supported |
| C55B16A | No representation clause for noncontiguous enumeration types |
| C86001F | package TEXT_IO uses package SYSTEM |
| C87B62A | Representation clause 'SIZE not supported |
| C87B62B | Representation clause 'STORAGE_SIZE not supported |
| C87B62C | Representation clause 'SMALL not supported |
| CA1012A | Separately compiled generic specifications and bodies not allowed |
| CA2009C | Separately compiled generic specifications and bodies not allowed |
| CA2009F | Separately compiled generic specifications and bodies not allowed |
| CA3004E | Inline pragmas ignored |
| CA3004F | Inline pragmas ignored |
| CE2201D | Instantiation of I/O package with unconstrained array type not allowed |

| | |
|---|---|
| CE2201E | Instantiation of I/O package with unconstrained array type not allowed |
| CE2401D | Instantiation of I/O package with unconstrained array type not allowed |
| EA3004C | Inline pragmas ignored |
| EA3004D | Inline pragmas ignored |
| LA3004A | Inline pragmas ignored |
| LA3004B | Inline pragmas ignored |

## 2.4 Implementation Characteristics

One of the purposes of validation is to determine the behavior of a compiler in those areas of the Ada Standard that permit implementations to differ. Class D and E tests specifically check for such implementation differences. However, inapplicable tests in other classes also characterize an implementation. This compiler is characterized by the following interpretations of the Ada Standard:

- Non-graphic characters.

  Non-graphic characters are defined in the ASCII character set but are not permitted in Ada programs, even within character strings. The compiler correctly recognizes these characters as illegal in Ada compilations. The characters are contained in the output listing but are not visible if printed except for some of the format effectors which have a visible effect.

- Capacities.

  The compiler correctly processes compilations containing loop statements nested to 65 levels, block statements nested to 65 levels, procedures nested to 17 levels, and 723 variables.

- Universal integer calculations.

  An implementation is allowed to reject universal integer calculations having values that exceed SYSTEM.MAX_INT. This implementation does not reject such calculations and processes them correctly.

- Universal real calculations.

  An implementation is allowed to reject universal real calculations having values that exceed certain precisions. This implementation does not reject such calculations and processes them correctly.

- Predefined types.

  This implementation supports the predefined type LONG_INTEGER. It does not support any other predefined numeric types except those required by the language.

- Based literals.

  An implementation is allowed to reject a based literal with value exceeding SYSTEM.MAX_INT during compilation or it may raise NUMERIC_ERROR during execution. This compiler raises NUMERIC_ERROR during execution.

- Array types.

  An implementation is allowed to raise NUMERIC_ERROR for an array having a 'LENGTH that exceeds STANDARD.INTEGER'LAST and/or SYSTEM.MAX_INT. When an array type is declared with an index range exceeding INTEGER values and with a component that is a null BOOLEAN array, this compiler does not raise any exceptions.

  When an array type is declared with an index range exceeding SYSTEM.MAX_INT values and with a component that is a null BOOLEAN array, this compiler does not raise any exceptions.

  A packed BOOLEAN array of length INTEGER_LAST+3 and a packed two-dimensional BOOLEAN array with INTEGER_LAST+3 components may be declared and used without raising exceptions.

  Null arrays with one dimension of length exceeding INTEGER'LAST may be declared and assigned without raising exceptions.

  In assigning one-dimensional array types, the entire expression is evaluated before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is comptaible with the target's subtype. In assigning two-dimensional array types, the entire expression is not evaluated before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. In assigning record types with discriminants, the entire expression is evaluated before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype.

- Discriminated types.

  An incompletely declared type with discriminants may be used in an access type definition and constrained either there or in later subtype indications.

- Aggregates.

  When evaluating the choices of a multi-dimensional aggregate the order in which choices are evaluated and index subtype checks are made depends upon the aggregate itself.

  When evaluating an aggregate containing subaggregates, not all choices are evaluated before being checked for identical bounds.

- Representation clauses.

  'SMALL length clauses are not supported.

  Enumeration representation clauses are not supported.

- Generics

  When given a separately compiled generic declaration some illegal instantiations and a body, the compiler rejects the body because it is not in the same compilation as its declaration.

- Package CALENDAR.

  TIME_OF and SPLIT are inverses when SECONDS is a non-model number.

- Pragmas.

  Pragma INLINE is not supported for procedures. It is not supported for functions.

- Input/output.

  Package SEQUENTIAL_IO cannot be instantiated with unconstrained array types and record types with discriminants. Package DIRECT_IO cannot be instantiated with unconstrained array types and record types with discriminants without defaults.

  More than one internal file can be associated with each external file for sequential I/O for both reading and writing. An external file associated with more than one internal file can be deleted.

  More than one internal file can be associated with each external file for direct I/O for both reading and writing. An external file associated with more than one internal file can be deleted.

  More than one internal file can be associated with each external file for text I/O for both reading and writing. An external file associated with more than one internal file can be deleted.

  An existing text file can be opened in OUT_FILE mode, can be created in OUT_FILE mode, and can be created in IN_FILE mode.

  Dynamic creation and resetting of a sequential file is allowed.

Temporary sequential files are given a name. Temporary direct files are given a name. Temporary files given names are not deleted when they are closed. They are however deleted when the Ada program that created them stops execution.

# CHAPTER 3

## Compiler Anomalies and Nonconformances

### 3.1 Anomalies

An anomaly is a test result that, given the pre-validation analysis, was not expected during formal validation but which is judged allowable by the AVF and the AVO under the circumstances of the validation. No anomalies were detected in this validation attempt.

### 3.2 Nonconformances

Any discrepancy between expected test results and actual test results is considered to be a nonconformance. No nonconformances were detected in this validation attempt.

# CHAPTER 4

# ADDITIONAL TESTING INFORMATION

## 4.1 Pre-Validation

Prior to validation, a set of test results for ACVC 1.7 produced by TeleSoft_Ada, Version 3.7, was submitted to IABG m.b.H., Dept SZT by the applicant for pre-validation review. Analysis of these results demonstrated that the compiler successfully passed all applicable tests.

## 4.2 Test Site

Tests were compiled and executed at Nynäshamn on two identical computers.

## 4.3 Test Tape Information

A test tape containing ACVC Version 1.7 was taken on-site by the validation team. This tape contained all tests applicable to this validation as well as all tests inapplicable to this validation. Tests that make use of values that are specific to the implementation were customized. The test suite was read from a tape. Files were structured into directories according to LRM chapters and test categories.

## 4.4 Testing Logistics

Once all tests had been loaded to disk, processing was begun using a test driver routine provided by the applicant. The test driver, a UNIX shell program, initiates the commands to compile, link, and execute Ada programs dynamically while processing the file directory.

The compiler supports various options that control its operation. The compiler was tested with the following option settings.

## Compiler Option Information

### Basic Information

The compilation process consists of 5 definite passes or steps:

Front-End:
Language control, translation to intermediate High-Form.

Middle Pass:
Resolution of tasking, translation to intermediate Low Form.

Code Generator:
Generation of native code for the target.

Prelinker:
Generation of elaboration code for main program.

Linker:
The native UNIX linker (ld) is used to link the object modules generated by the compiler.

Below the switches of the different passes are tabulated. The switches pertinent to the UNIX linker are however not treated here.

**Front-End Switches**

| Name | (default) | Effect |
|---|---|---|
| Error_File | (+) | Error file |
| To_Output | (+) | List to standard output |
| Dots | (-) | Display one dot per line scanned |
| Comments | (-) | Pass comments to sematics |
| Notify | (+) | Ask user for action at error |
| Verbose | (+) | Verbose output |
| Long_Integers | (-) | Declare Long_Integer when generating package standard. |
| 32bit_Integers | (-) | Define type Integer with 32 bits when generating package standard. |
| Long_Float | (-) | Declare Long_Float when generating package standard. |
| D_Format | (-) | Define type Long_Float with DEC d_format when generating package standard. |
| Fe_Max_Pages | 1500 | Pages in memory |
| Context | 1 | Lines of context in errors |
| Recovery_Level | 50 | Tokens of context in error recovery |
| Message_Level | 0 | Verboseness of error messages |

(to be continued)

| Front-End Switches (continued) | | |
|---|---|---|
| Name | (default) | Effect |
| Abort_Syntax_Count | 999 | Max # before aborting compiler |
| Abort_Warning_Count | 999 | Max # before aborting compiler |
| Abort_Semantic_Count | 999 | Max # before aborting compiler |
| Quit_Syntax_Count | 999 | Max # before aborting compiler |
| Quit_Warning_Count | 999 | Max # before aborting compiler |
| Quit_Semantic_Count | 999 | Max # before aborting compiler |
| Warnings | (+) | Display warning messages |
| Fe_Debug | "" | Debug set |
| Console_Name | "<stdout>:" | Name of Console |
| Console_Form | "" | Form of Console |
| Keyboard_Name | "<stdin>:" | Name of Keyboard |
| Keyboard_Form | "" | Form of Keyboard |
| Error_Name | "errors" | Name of list file |
| Error_Form | "" | Form of list file |
| Table_Name | "fe_code:ptable" | Name of parse table |
| Table_Form | "" | Form of parse table |
| Info_Name | "ainfo" | Name of Info file |
| Info_Form | "" | Form of Info file |
| Message_Name | "fe_code:mfile" | Name of error message file |
| Message_Form | "" | Form of error message file |
| Liblst_Name | "liblst" | Name of library file |
| Liblst_Form | "" | Form of library file |
| mpp | (-) | (obsolete) |
| mpp_trix | (-) | (obsolete) |
| lock | true | Controls updating of Ada library |
| Standard | (-) | Generate package Standard. |
| Source_Name | "" | Name of Source file |
| Source_Form | "" | Form of Source file |
| Help | (-) | Display this text |

| Middle Pass Switches | | |
|---|---|---|
| Name | (default) | Effect |
| check | (-) | Check for MPST_Node_Already_Exists |
| debug | (-) | Turn on debugging output |
| fe_debug | "" | Turn on debugging output for FE packages |
| liblst | "liblst" | Name of sublibrary list file |
| max_pages | 1000 | Virtual space pages (like fe_max_pages) |
| mp_verbose | (+) | Display version, copyright |
| standard | (-) | Compile standard |
| varc | "" | Various effects, depending on arguments |
| write | (-) | Write LF even if an error occurred |

| Code-Generator Switches | | |
|---|---|---|
| Name | (default) | Effect |
| d | (-) | Debug switch, generates debug info for debugger. |
| e | (-) | Generate execution profile information. |
| i | (+) | Interactive switch, asks action in case of error. |
| l=<name> | (-) | Creates an assemble list with desired name. |
| l | (-) | Creates an assemble list with default name ( .s). |
| lib=<name> | "liblst" | Defines the library list file. |
| mode=<mode> | "restore" | Determines the mode to open the library. |
| n<=name> | (-) | Creates labels with name <name>lineno for each line. |
| n | (-) | Creates labels with default names (S<line no>). |
| o=<name> | "<cmpu name>.o" | Create an object file with name <name>. |
| p=<digits> | "10240" | Use literal pool of size <digits>. |
| q | (-) | Do the job quietly. |
| r<=word> | (-) | Produce relative addresses. Default is absolute. |
| s | (+) | Give source info (line no) when reporting exceptions. |
| shadow | (+) | Variable shadowing desired. |
| slf | (-) | Check the code generator, prints SLF-instructions. |
| stack | (-) | Suppress checking on stack expansion for overflow. |
| t=<name> | (-) | Source file; listing with source lines desired. |
| v=<digits> | "1500" | Virtual memory pages size. |
| c | (-) | Use CHK instr when applicable. |
| m=<targeting> | "mc68000" | Choose target machine (MC68k family). |

| Prelinker Switches | | |
|---|---|---|
| Name | (default) | Effect |
| lib=<name> | "liblst" | Names the library file. |
| l | (+) | Produce link script. |
| o=<name> | "tsamain.o" | Names the main object file. |
| q | (-) | Do the job quietly. |
| r | (+) | Recompile inconsistent units. |
| v=<digits> | "1500" | Virtual memory pages size. |

**Options used in compiling ACVC tests:**

| Front_End | |
|---|---|
| -to_output | No list is generated |
| -notify | Continue on error |
| Fe_max_pages | 1500 |
| others | =default |

| Middle Pass | |
|---|---|
| +varc=pet2 | Controls allocation and alignment |
| -notify | Continue on error |
| max_pages | 1500 |
| others | =default |

| Code Gen and Prelinker | |
|---|---|
| -l | No assembly list generated (code gen) |
| -l | No link script generated (prelinker) |
| -i | No interactive execution |
| -r | No recompilation of inconsistent units (prelinker) |
| v | 1500 |
| others | =default |

Test were run on two machines simulataneously starting with the B-Tests. On each machine a batch queue was used. For each test a new program library was created containing the standard Ada package and the ACVC support packages and procedures. The report package was modified in order to print the date and time of test execution. For each chapter the compiler listings and the results, if any, were written to individual files. These files were written on tape in UNIX dd format and archived.

### 4.5 Testing Duration

The ACVC has not been designed for use in measuring compiler performance. However, information about the length of time needed to test the compiler may characterize compiler performance in processing a large number of programs.

Testing started in the afternoon of 86-04-11 and was completed in the morning of 86-04-20. Testing was interrupted twice for a total of about 20 hours.

# CHAPTER 5

## SUMMARY AND CONCLUSIONS

The IABG m.b.H., Dept SZT, identified 1993 of the 2286 tests in Version 1.7 of the Ada Compiler Validation Capability to be processed during the validation of TeleSoft Ada including 7 tests for the report package. Out of the 1993 tests, one was withdrawn because of test errors (15 tests were withdrawn previously), 28 were ruled inapplicable, and the other 1964 were passed by the compiler.

The IABG m.b.H., Dept SZT concludes that these results demonstrate acceptable conformance to the Ada Standard.

# APPENDIX A

## COMPLIANCE STATEMENT

The only allowed implementation dependencies correspond to implementation-dependent pragmas and attributes, to certain machine-dependent conventions as mentioned in Chapter 13 of MIL-STD-1815A, and to certain allowed restrictions on representation classes. The implementation-dependent characteristics of the TeleSoft_Ada are described in the following sections which discuss topics one through eight as stated in Appendix F of the Ada Language Reference Manual (ANSI/MIL-STD-1815A).

1. Implementation Dependent Pragmas

There is one implementation-defined pragma, COMMENT. It has the form:

```
pragma COMMENT( <string_literal> );
```

It may only appear within a compilation unit and has the effect of embedding the given sequence of characters in the object code of the compilation unit.

2. Implementation Dependent Attributes

There are no implementation dependent attributes.

3. Specification of Package SYSTEM

```
PACKAGE System IS

  SUBTYPE Byte is Natural range 0 .. 255;

  TYPE Address is ACCESS Integer;
  TYPE Subprogram_Value is PRIVATE;

  TYPE Name    IS (TeleSoft_Ada);

  System_Name  : CONSTANT name := TeleSoft_Ada;

  Storage_Unit : CONSTANT := 8;
  Memory_Size  : CONSTANT := (2 ** 24) - 1;

  -- System-Dependent Named Numbers:

  Min_Int      : CONSTANT := -(2 ** 31);
  Max_Int      : CONSTANT := (2 ** 31) - 1;
  Max_Digits   : CONSTANT := 6;
  Max_Mantissa : CONSTANT := 31;
  Fine_Delta   : CONSTANT := 1.0 / (2 ** (Max_Mantissa - 1));
  Tick         : CONSTANT := 10.0E-3;

  -- Other System-Dependent Declarations

  SUBTYPE Priority IS Integer RANGE 0 .. 63;

  Max_Object_Size   : CONSTANT := Max_Int;
  Max_Record_Count  : CONSTANT := Max_Int;
  Max_Text_Io_Count : CONSTANT := Max_Int - 1;
  Max_Text_Io_Field : CONSTANT := 1000;

PRIVATE
  TYPE Subprogram_Value is
    record
```

```
        Proc_addr    : Address;
        Static_link  : Address;
        Global_frame : Address;
    end record;
END System;
```

4. Restrictions on Representation Clauses

The compiler supports the following representation clauses:

Length Clauses: for tasks 'STORAGE_SIZE (LRM 13.2(C)).
Address Clauses: for objects and entries (LRM 13.5)

5. Implementation dependent naming conventions

There are no implementation-generated names denoting implementation dependent components.

6. Interpretation of expressions in address clauses

Expressions that appear in address specifications are interpreted as the first storage unit of the object.

7. Restrictions on Unchecked Conversions

Unchecked conversions are allowed between variables of types (or subtypes) T1 and T2 provided that 1) they have the same static size, 2) they are not unconstrained array types, and 3) they are not private (unless they are subtypes of or are derived from type SYSTEM.ADDRESS).

8. I/O Package Characteristics

Instantiations of DIRECT_IO and SEQUENTIAL_IO are supported with the following exceptions,

- o unconstrained array types
- o unconstrained types with discriminants without default values.

Calling CREATE with a name of an existing external file does not raise an exception (the old file is overwritten).

In DIRECT_IO the type COUNT is defined as follows:

```
type COUNT is range 0 .. 16#7FFFFFFF#;
```

In TEXT_IO the type COUNT is defined as follows:

```
type COUNT is range 0 .. 16#7FFFFFFD#;
```

In TEXT_IO the subtype FIELD is defined as follows:

```
type FIELD is INTEGER range 0 .. 1000;
```

9. Package standard

The package standard used for this compiler corresponds to the specification given in LRM Appendix C. The implementation dependent parts are implemented as follows:

```
type integer is range -32768..32767;

type long_integer is range -2147483648..2147483647;

type float is digits 6 range

        -2#0.1....1#E128..2#0.1....1#128;
           24              24

type duration is delta 2#1#E-14 range -86400..86400.0;
```

10. File naming conventions

There are no restrictions on file names other than those imposed by the operating system.

## APPENDIX B

## TEST PARAMETERS

Certain tests in the ACVC make use of implementation-dependent values, such as the maximum length of an input line and invalid file names. A test that makes use of such values is identified by the extension .TST in its file name. Actual values to be substituted are identified by names that begin with a dollar sign. A value is substituted for each of these names before the test is run. The values used for this validation are given below.

| | |
|---|---|
| $MAX_IN_LEN | 200 |
| $BIG_ID1 | String(1..200) := (1..199 => 'A', 200 => '1') |
| $BIG_ID2 | String(1..200) := (1..199 => 'A', 200 => '2') |
| $BIG_ID3 | String(1..200) := (1..100 => 'A', 101 => '3', 102..200 => 'A') |
| $BIG_ID4 | String(1..200) := (1..100 => 'A', 101 => '4', 102..200 => 'A') |
| $NEG_BASED_INT | 16#FFFFFFFE# |
| $BIG_INT_LIT | String(1..200) := (1..197 => '0', 198..200 => "298") |
| $BIG_REAL_LIT | String(1..200) := (1..194 => '0', 195..200 => "69.0E1") |
| $EXTENDED_ASCII_CHARS | "abcdefghijklmnopqrstuvwx yz!$%?@[\]^`{}~" |
| $NON_ASCII_CHAR_TYPE | (NON_NULL) |
| $BLANKS | String(1..180) := (1..180 => ' ') |
| $MAX_DIGITS | 6 |
| $NAME | No such numeric type - long integer used |
| $INTEGER_FIRST | -2**31; |
| $INTEGER_LAST | 2**31-1; |
| $MAX_INT | 2**31-1; |
| $LESS_THAN_DURATION | -86_401.0 |
| $GREATER_THAN_DURATION | 86_401.0 |
| $LESS_THAN_DURATION_BASE_FIRST | -131_072.0 |
| $GREATER_THAN_DURATION_BASE_LAST | 131_072.0 |
| $COUNT_LAST | 2**31-2; |
| $FIELD_LAST | 1000 |
| $FILE_NAME_WITH_BAD_CHARS | "X}]!/@#%$^&~Y" |
| $FILE_NAME_WITH_WILD_CARD_CHAR | "XYZ*" |
| $ILLEGAL_EXTERNAL_FILE_NAME1 | "BAD-CHARACTER%/*" |
| $ILLEGAL_EXTERNAL_FILE_NAME2 | "AAA/AAA" |

# APPENDIX C

## COMMAND SCRIPTS

Command scripts used for this validation were reviewed by the validation team, but are not released for publication.

END

4-87

DTIC